Abschätzung der Verhältnisse von aufeinander folgenden hochzusammengesetzten Zahlen

Jan Behrens

25. August 2008
Korrektur vom 18. Juli 2009

Die Teileranzahl $d(n)$ einer Zahl n ist die Anzahl von Zahlen, durch die sich die Zahl n (restfrei) teilen lässt. Eine Zahl n ist dann eine hochzusammengesetzte Zahl, wenn ihre Teileranzahl $d(n)$ größer ist, als die jeweiligen Teileranzahlen aller natürlichen Zahlen kleiner als n. Zum Beispiel ist 12 eine hochzusammengesetzte Zahl, da sie 6 Teiler hat (nämlich 1,2,3,4,6 und 12), und die Zahlen 1,2,3,4,5,6,7,8,9,10 oder 11 alle jeweils weniger als 6 Teiler besitzen.

Der englische Begriff für hochzusammengesetzte Zahlen lautet „highly composite numbers". Ich werde die hochzusammengesetzten Zahlen daher im folgenden als HCNs bezeichnen.

Im folgenden soll bewiesen werden, dass es nur 8 HCNs gibt, deren direkt nachfolgende HCN größer als das jeweils $\frac{3}{2}$-fache ist. Es gibt einen ähnlichen Beweis, der zeigt, dass es nur 7 HCNs gibt (Ganzzahl-Folge A072938), deren jeweils nachfolgende HCN doppelt so groß ist.

Habe die natürliche Zahl n eine Primfaktorzerlegung von

$$n = \prod_{p \in \mathbb{P}} p^{c_p}$$

dann lässt sich die Teileranzahl $d(n)$ sehr einfach mit Hilfe der Formel

$$d(n) = \prod_{p \in \mathbb{P}} (c_p + 1)$$

bestimmen. Betrachten wir nun eine HCN der Form

$$n_{\mathrm{A}} = 2^{c_2} \cdot 3^{c_3} \cdot \prod_{p \in 5,7,11,\ldots} p^{c_p}$$

dann finden wir eine um den Faktor $\frac{3}{2}$ größere Zahl der Form

$$n_{\mathrm{B}} = 2^{c_2-1} \cdot 3^{c_3+1} \cdot \prod_{p \in 5,7,11,\ldots} p^{c_p}$$

mit mehr Teilern genau dann, wenn $d(n_{\mathrm{B}}) > d(n_{\mathrm{A}})$ gilt, also wenn

$$\begin{array}{rcl} c_2 \cdot (c_3+2) \cdot \prod\limits_{p \in 5,7,11,\ldots} (c_p+1) & > & (c_2+1) \cdot (c_3+1) \cdot \prod\limits_{p \in 5,7,11,\ldots} (c_p+1) \\ c_2 \cdot (c_3+2) & > & (c_2+1) \cdot (c_3+1) \\ c_2c_3 + 2c_2 & > & c_2c_3 + c_2 + c_3 + 1 \\ c_3 & < & c_2 - 1 \end{array}$$

ist. Eine zur HCN n_{A} um den Faktor $\frac{4}{3}$ größere Zahl der Form

$$n_{\mathrm{C}} = 2^{c_2+2} \cdot 3^{c_3-1} \cdot \prod_{p \in 5,7,11,\ldots} p^{c_p}$$

mit mehr Teilern findet sich genau dann, wenn $d(n_{\mathrm{C}}) > d(n_{\mathrm{A}})$ gilt, also wenn

$$\begin{array}{rcl} (c_2+3) \cdot c_3 \cdot \prod\limits_{p \in 5,7,11,\ldots} (c_p+1) & > & (c_2+1) \cdot (c_3+1) \cdot \prod\limits_{p \in 5,7,11,\ldots} (c_p+1) \\ (c_2+3) \cdot c_3 \cdot & > & (c_2+1) \cdot (c_3+1) \\ c_2c_3 + 3c_3 & > & c_2c_3 + c_2 + c_3 + 1 \\ 2c_3 & > & c_2 + 1 \end{array}$$

Die beiden Ungleichungen

$$\begin{array}{rcl} c_3 & < & c_2 - 1 \\ 2c_3 & > & c_2 + 1 \end{array}$$

sind lediglich für $(c_2, c_3) \in S$ mit

$$S = \{(0,0), (0,1), (1,1), (2,1), (3,2)\}$$

nicht erfüllt, d.h. zu jeder HCN n_{D} deren Primfaktorzerlegung $(c_2, c_3) \notin S$ ergibt, findet sich eine um den Faktor $\frac{3}{2}$ oder $\frac{4}{3}$ größere Zahl mit mehr Teilern. Hieraus folgt, dass die auf n_{D} direkt nachfolgende HCN maximal um den Faktor $\frac{3}{2}$ größer sein kann. HCNs deren nachfolgende HCN um mehr als $\frac{3}{2}$ größer ist, müssen somit eine Primfaktorzerlegung mit $(c_2, c_3) \in S$ aufweisen. Im Folgenden wird die Untersuchung aller Zahlen mit einer solchen

Primfaktorzerlegung zeigen, dass nur endlich viele HCNs, nämlich 1, 2, 6, 12, 60, 360, 2520 sowie 27720 eine direkt nachfolgende HCN besitzen, die um mehr als das $\frac{3}{2}$-fache größer ist.

Mit Hilfe eines Computerprogrammes lassen sich die ersten 35 HCNs leicht berechnen: 1, 2, 4, 6, 12, 24, 36, 48, 60, 120, 180, 240, 360, 720, 840, 1260, 1680, 2520, 5040, 7560, 10080, 15120, 20160, 25200, 27720, 45360, 50400, 55440, 83160, 110880, 166320, 221760, 277200, 332640, 498960.

Wir betrachten zunächst den einfachen Fall $(c_2, c_3) = (0, 0)$. Primfaktorexponenten von HCNs müssen mit wachsender Primzahl monoton fallend sein, da man ansonsten durch Vertauschen der Exponenten eine kleinere Zahl mit genauso vielen Teilern erhalten würde. Da $c_3 = 0$ ist, müssen also auch $c_5 = c_7 = c_{11} = ... = 0$ ein. Die sich daraus ergebende Zahl $2^{c_2} \cdot 3^{c_3} = 2^0 \cdot 3^0 = 1$ ist die erste HCN deren direkt nachfolgende HCN 2 doppelt so groß ist.

Für den Fall $(c_2, c_3) = (1, 0)$ sind ebenfalls $c_5 = c_7 = c_{11} = ... = 0$. Die sich daraus ergebende Zahl $2^{c_2} \cdot 3^{c_2} = 2^1 \cdot 3^0 = 2$ ist die zweite HCN, deren direkt nachfolgende HCN 4 doppelt so groß ist.

Für den Fall $(c_2, c_3) = (1, 1)$ lässt sich keine triviale Aussage mehr über die Exponenten c_5, c_7, c_{11}, u.s.w. treffen. Die HCN $2^1 \cdot 3^1 = 6$ ist die dritte HCN, deren direkt nachfolgende HCN 12 doppelt so groß ist. Für jede Zahl $2^1 \cdot 3^1 \cdot 5^1 \cdot \prod_{p=7,11,...} p^{c_p}$ können wir allerdings eine Zahl $2^2 \cdot 3^2 \cdot 5^0 \cdot \prod_{p=7,11,...} p^{c_p}$ finden, die um das $\frac{6}{5}$-fache größer ist und mehr Teiler besitzt, da $(2+1) \cdot (2+1) \cdot (0+1) > (1+1) \cdot (1+1) \cdot (1+1) \Leftrightarrow 9 > 8$. Weitere Möglichkeiten (mit $c_5 > 1$) entfallen aufgrund der oben erwähnten Monotonie der Primfaktorexponenten c_p.

Beim Betrachten des Falles $(c_2, c_3) = (2, 1)$ ergibt sich die HCN $2^2 \cdot 3^1 = 12$ als vierte, und die HCN $2^2 \cdot 3^1 \cdot 5^1 = 60$ als fünfte HCN, deren direkt nachfolgende HCNs 24 und 120 doppelt so groß sind. Die Zahl $2^2 \cdot 3^1 \cdot 5^1 \cdot 7^1 = 420$ ist keine HCN und zu Zahlen der Form $2^2 \cdot 3^1 \cdot 5^1 \cdot 7^1 \cdot 11^1 \cdot \prod_{p=13,17...} p^{c_p}$ findet sich immer eine Zahl $2^4 \cdot 3^2 \cdot 5^1 \cdot 7^1 \cdot 11^0 \cdot \prod_{p=13,17...} p^{c_p}$, die um das $\frac{12}{11}$-fache größer ist und mehr Teiler besitzt, da $(4+1) \cdot (2+1) \cdot (1+1) \cdot (1+1) \cdot (0+1) > (2+1) \cdot (1+1) \cdot (1+1) \cdot (1+1) \cdot (1+1) \Leftrightarrow 60 > 48$. Weitere Möglichkeiten (mit $c_5 > 1$, $c_7 > 1$ oder $c_{11} > 1$) entfallen aufgrund der Monotoniebedigung der Primfaktorexponenten.

Nun betrachten wir den letzten Fall $(c_2, c_3) = (3, 2)$. Die Zahl $2^3 \cdot 3^2 = 72$ ist keine HCN, die Zahlen $2^3 \cdot 3^2 \cdot 5^1 = 360$ und $2^3 \cdot 3^2 \cdot 5^1 \cdot 7^1 = 2520$ sind die sechste und siebte HCN deren direkt nachfolgende HCNs 720 und 5040 doppelt so groß sind. Mit der Zahl $2^3 \cdot 3^2 \cdot 5^1 \cdot 7^1 \cdot 11^1 = 27720$ haben wir eine HCN gefunden deren direkt nachfolgende HCN zwar größer ist als $\frac{3}{2}$, jedoch nicht um das doppelte, sondern um $\frac{18}{11}$. Die Zahl $2^3 \cdot 3^2 \cdot 5^1 \cdot 7^1 \cdot 11^1 \cdot 13^1 = 360360$

ist keine HCN, und zu jeder Zahl $2^3 \cdot 3^2 \cdot 5^1 \cdot 7^1 \cdot 11^1 \cdot 13^1 \cdot 17^1 \cdot \prod_{p=19,23,\ldots} p^{c_p}$ lässt sich immer eine Zahl $2^6 \cdot 3^3 \cdot 5^1 \cdot 7^1 \cdot 11^1 \cdot 13^1 \cdot 17^0 \cdot \prod_{p=19,23,\ldots} p^{c_p}$ finden, die $\frac{24}{17}$-mal größer ist und mehr Teiler besitzt, da $(6+1)\cdot(3+1)\cdot(1+1)\cdot(1+1)\cdot(1+1)\cdot(1+1)\cdot(0+1) > (3+1)\cdot(2+1)\cdot(1+1)\cdot(1+1)\cdot(1+1)\cdot(1+1)\cdot(1+1) \Leftrightarrow 448 > 384$. Für alle Zahlen der Form $2^3 \cdot 3^2 \cdot 5^2 \cdot \prod_{p=7,11,\ldots} p^{c_p}$ gibt es immer eine um den Faktor $\frac{6}{5}$ größere Zahl $2^4 \cdot 3^3 \cdot 5^1 \cdot \prod_{p=7,11,\ldots} p^{c_p}$ mit mehr Teilern, da $(4+1)\cdot(3+1)\cdot(1+1)\cdot(0+1) > (3+1)\cdot(2+1)\cdot(2+1) \Leftrightarrow 40 > 36$. Alle weiteren Möglichkeiten (z.B. $c_5 > 2$) entfallen wegen der Monotoniebedingung.

Es konnte somit gezeigt werden, dass das Verhältnis zweier aufeinander folgender HCNs bis auf 8 Einzelfälle nie größer als $\frac{3}{2}$ ist. In 7 dieser Fälle handelt es sich um HCNs deren jeweils direkt nachfolgende HCN genau doppelt so groß ist, der 8. Fall jedoch zeigt auf, dass es genau eine HCN gibt, deren direkt nachfolgende HCN mehr als $\frac{3}{2}$-mal jedoch nicht 2 mal so groß ist. Diese HCN lautet 27720; ihr Nachfolger lautet 45360.

Es wurde folgendes Haskell-Programm für die Berechnungen der HCNs verwendet:

```
import Data.Ratio
import Data.Set (Set)
import qualified Data.Set as Set

printList :: (Show a) => [a] -> IO()
printList  =  putStr . concat . map (\x -> show x ++ "\n")

isPrime n
  | n >= 2     =  all isNotDivisor $ takeWhile smallEnough primes
  | otherwise  =  False
  where
    isNotDivisor d  =  n `mod` d /= 0
    smallEnough d   =  d^2 <= n

primes  =  2 : filter isPrime [ 2 * n + 1 | n <- [1..] ]

primeSynthesis  =  partialSynthesis 1 primes
  where
    partialSynthesis n _ []           =  n
    partialSynthesis n (p:ps) (c:cs)  =  partialSynthesis (n * p^c) ps cs
```

```
primeAnalysis n
  | n < 1   =  undefined
  | n == 1  =  []
  | n > 1   =  reverse $ buildPrimeCounts [0] n
  where
    buildPrimeCounts (c:cs) n
      | n == 1          =  (c:cs)
      | n `mod` p == 0  =  buildPrimeCounts (c+1 : cs) (n `div` p)
      | otherwise       =  buildPrimeCounts (0:c:cs) n
      where  p  =  primes !! (length cs)

divisorCount n  =  product $ map (+1) $ primeAnalysis n

primorialProducts  =  resFrom 1
  where
    resFrom n             =  resBetween n (4*n - 1) ++ resFrom (4*n)
    resBetween start end  =  Set.toAscList $ Set.fromList $ unorderedList
      where
        unorderedList  =  filter (>= start) (1 : build 0 [])
        build pos exponents
          | nextNumber <= end  =  nextNumber : build 0 nextCombination
          | newPrime           =  []
          | otherwise          =  build (pos + 1) exponents
          where
            newPrime  =  pos >= length exponents
            nextCombination
              | newPrime   =  replicate (length exponents + 1) 1
              | otherwise  =  replicate (pos + 1) ((exponents !! pos) + 1)
                              ++ drop (pos + 1) exponents
            nextNumber  =  primeSynthesis nextCombination

filterStrictlyMonotonicDivisorCount  =  filterRest 0
  where
    filterRest _ []  =  []
    filterRest lim (num:nums)
      | divisorCount num > lim  =  num : filterRest (divisorCount num) nums
      | otherwise               =  filterRest lim nums
```

```
highlyCompositeNumbers
  =  filterStrictlyMonotonicDivisorCount primorialProducts

findBigGaps []   =  []
findBigGaps [_]  =  []
findBigGaps (x1:x2:xs)
  | x1 * 3 < x2 * 2  =  (x1,x2) : findBigGaps (x2:xs)
  | otherwise        =  findBigGaps (x2:xs)

findGaps []   =  []
findGaps [_]  =  []
findGaps (x1:x2:xs)
  | x1 * 3 <= x2 * 2  =  (x1,x2) : findGaps (x2:xs)
  | otherwise         =  findGaps (x2:xs)
```